designers, developers can incorporate the spreadsheet in a graphical user interface, e.g., Visual Basic as a development platform. The simulation spreadsheet is usually invisible and populated using functions provided by the engine. It is very important that all modifications that the ICA needs to know about go through the engine because only the engine knows how to call the ICA. This significantly reduced the skill level required from programmers, and greatly reduced the time required to program each task. In addition, the end-product was less prone to bugs, because the tutor management was centralized. If there was a tutor problem, we only had to check on section of code. Finally, since the simulation engine loaded the data from a spreadsheet, the chance of data inconsistency between the tutor and the application was nil.

Figure **25** is a block diagram setting forth the architecture of a simulation model in accordance with a preferred embodiment. The Simulation Object Model consists of a spreadsheet model, a spreadsheet control object, a simulation engine object, a simulation database, input objects, output objects, list objects and path objects. The first object in our discussion is the Spreadsheet object. The Spreadsheet is the support for all simulation models. A **control** object that is readily integrated with the Visual Basic development plat. The control supports printing and is compatible with Microsoft Excel spreadsheets. With that in mind, designers can use the power of Excel formulas to build the simulation. The different cells contained in the spreadsheet model can be configured as Inputs, Outputs or Lists and belong to a simulation Path. All cells in the spreadsheet that need to be manually entered by the designer or the student via the GBS application are represented by input objects. Every input has the following interface:

Field Name	Data Type	Description
InputID	long	Primary Key for the table
TaskID	long	TaskID of the task associated with the input
PathID	long	PathID of the path associated with the input
InputName	string*50	Name of the input
InputDesc	string*255	Description of the input
ReferenceName	string*50	Name of the spreadsheet cell associated with the input
TutorAware	boolean	Whether the ICA should be notified of any changes to the input
SourceItemID	long	SourceItemID if input is a distinct input; 0 if input is a drag drop input
TargetID	long	TargetID of the input
Row	long	Spreadsheet row number of the input → speed optimization
Column	long	Spreadsheet column number of the input → speed optimization
SheetName	string*50	Sheet name were the input is located → speed optimization

This information is stored for every input in the Input table of the simulation database (ICASim.mdb). Refer to the example below. When designers construct their simulation model, they must be aware of the fact that there are 2 types of

Inputs: Distinct Input & Drag & Drop Input. The Distinct Input consists of a single spreadsheet cell that can be filled by the designer at design time or by the GBS application at run time via the simulation engine object's methods. The purpose of the cell is to provide an entry point to the simulation model. This entry point can be for example an answer to a question or a parameter to an equation. If the cell is TutorAware (all inputs are usually TutorAware), the ICA will be notified of any changes to the cell. When the ICA is notified of a change two messages are in fact sent to the ICA: An ICANotifyDestroy message with the input information i.e., SourceItemID, TargetID and null as Attribute. This message is to advise the ICA to remove this information from its memory. An ICANotifyCreate message with the input information i.e., SourceItemID, TargetID, Attribute (cell numeric value) . This message is to advise the ICA to add this information to its memory. A Distinct Input never requires that a user answer a mathematics question.

These are the steps required to configure that simulation: Define a name for cell C2 in Excel. Here we have defined "Distinct_Input". In the ICA, define a task that will be assigned to the simulation. Ex: a TaskID of 123 is generated by the ICA. In the ICA, define a Target for the input. Ex: a TargetID of 4001 is generated by the ICA. In the ICA, define a SourceItem for the input. Ex: a SourceItemID of 1201 is generated by the ICA. Associate the input to a path (refer to Path object discussion). Add the information in the Input table of the simulation engine database. A record in an Input table is presented below.

InputID:	12345
TaskID:	123
PathID:	1234
InputName:	Question 1 input
InputDesc:	Distinct input for Question 1
ReferenceName:	Distinct_Input
TutorAware:	True
SourceItemID	1201
TargetID:	4001
Row:	2
Column:	3
SheetName:	Sheet1

The Row, Column and SheetName are filled in once the user clicks "Run Inputs/Outputs". The simulation engine decodes the defined name (Reference Name) that the designer entered, and populates the table accordingly. This is an important step. We had several occasions when a designer would change the layout of a spreadsheet, i.e., move a defined name location, then forget to perform this step. As such, bizarre data was being passed to the tutor; whatever data happened to reside in the old row and column. Once the configuration is completed, the designer can now utilize the ICA Utilities to test the simulation.

The drag & drop input consist of two consecutive spreadsheet cells. Both of them have to be filled by the designer at design time or by the GBS application at run time via the simulation engine object's methods. This type of input is used usually when the user must choose one answer among a selection of possible answers. Drag & drop inputs are always TutorAware. The

left most cell contains the SourceItemID of the answer picked by the user (every possible answer needs a SourceItemID) and the rightmost cell can contain a numeric value associated to that answer. You need to define a name or ReferenceName in the spreadsheet for the rightmost cell. ICA will be notified of any changes to either one of the cells. When the ICA is notified of a change two messages are in fact sent to the ICA: An ICANotifyDestroy message with the input information i.e., SourceItemID before the change occurred, TargetID of the input and the Attribute value before the change occurred. An ICANotifyCreate message with the input information i.e., SourceItemID after the change occurred, TargetID of the input and the Attribute value after the change occurred.

These are the steps required to configure that section of the simulation: Define a name for cell C11 in Excel. Here we have defined "DragDrop_Input"; Let's use the same TaskID as before since Question 2 is part of the same simulation as Question 1. Ex: TaskID is 123; In the ICA, define a Target for the input. Ex: a TargetID of 4002 is generated by the ICA; In the ICA, define a SourceItem for every possible answer to the question. Ex: SourceItemIDs 1202 to 1205 are generated by the ICA; Associate the input to a path (refer to Path object discussion); and Add the information in the Input table of the simulation engine database. A record in the Input table in accordance with a preferred embodiment is presented below.

InputID:	12346
TaskID:	123
PathID.	1234
InputName:	Question 2 input
InputDesc:	Drag & Drop input for Question 2
ReferenceName:	DragDrop_Input
TutorAware:	True
SourceItemID	0 ***
TargetID:	4002
Row:	11
Column:	3
SheetName.	Sheet1

The list object consists of one cell identifying the list (cell #1) and a series of placeholder rows resembling drag & drop inputs (cells #1.1 - 1.n to cells #n.1- n.n). The list is used usually when the user must choose multiple elements among a selection of possible answers. Cell #1 must have a uniquely defined name also called the list name. Cells #1.1 to #n.1 can contain the SourceItemID of one possible answer picked by the user (every possible answer needs a SourceItemID). The content of these cells must follow this format : ~ListName~SourceItemID. Cells #1.2 to #n.2 will hold the numeric value (attribute) associated with the SourceItemID in the cell immediately to the left. Cells #1.3 - 1.n to #n.3 - n.n are optional placeholders for data associated with the answer. KEY NOTE: When implementing a list object the designer must leave all the cells under #n.1 to #n.n blank because this range will shift up every time an item is removed from the list.